U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
SYSTEMS DEVELOPMENT OFFICE
TECHNIQUES DEVELOPMENT LABORATORY

TDL OFFICE NOTE 79-13

TECHNIQUES DEVELOPMENT LABORATORY SOFTWARE STANDARDS

Harry R. Glahn

June 1979

TECHNIQUES DEVELOPMENT LABORATORY

SOFTWARE STANDARDS

1. INTRODUCTION

In the early days of TDL, much of the computer work was carried out by individuals with little or no interaction with others. No large projects existed in which software was developed as a group effort. Almost never were programs documented so that others might know of their existence or how to use them. Of course, no software standards existed.

Gradually, larger tasks and projects were formed and these were maintained for longer periods of time. Software documentation began to appear sporadically. Finally in 1972, a TDL library of programs was established and standards for documentation prescribed.

Today, much of the work in TDL is interrelated, the largest common factor being the MOS Development System (Glahn, 1975) and its related implementation programs. A library of about 100 members is maintained for use on the IBM 360/195 and documentation is provided (Glahn, et al., 1975). Its use is expanding, and more people are contributing to the program library. The development system data bases are being increased, both in length and types of data, making expanded use of the system more attractive.

A few weeks ago we took delivery of a dual Eclipse computer system. This equipment will eventually be used at the National Meteorological Center (NMC) to produce automated aviation terminal forecasts (FT's). It will also be available for testing new concepts in producing automated forecasts on AFOS (Automation of Field Operations and Services) equipment at Weather Service Forecast Offices (WSFO's).

I believe it is very important that we establish software standards for use within TDL. Reasons for standards include:

- With more interaction among programmers and more use being made of our MOS system, it is important that certain guidelines be followed so that one person can "read" another person's program.

- Many times it falls to someone other than the originator to modify or maintain a program. Again, if the program has been written and documented according to prescribed rules, revisions and maintenance are much easier.

- In the future, we may have to contract for a larger portion of our software. The contractor may deliver the program and then not be available to maintain it. Uniformity of appearance and other standardization will increase the ease of in-house or other contractor's understanding and modifying the code.

- Standardization will reduce errors in coding and punching (keystroking). The eye and mind become accustomed to "patterns," and a break in pattern may be an error. If there are no patterns in the code, this human error detection feature cannot operate. Also, if a variable starting with I, J, K, L, M, or N is never real (floating point), then mixed mode errors etc. are much easier to detect.

- Interfacing one set of programs with another is many times much easier if each has been written according to the same standards.

- Converting software from one computer system to another is easier if it is all written to the same standards.

- Persons writing code and having it punched by others need not explain a preferred format; it will be defined in writing for the card puncher to follow.

- New employees with little or no programming experience can be more easily trained in good procedures if they are written down and everyone in the Lab follows them.

- Some simple optimization procedures, if followed, can reduce execute time considerably. However, the primary purpose for TDL standards is not cp optimization. Also, what is optimum for one system might not be for another.

In summary, the objectives of the TDL standards are to enhance clarity, testability, maintainability, and person-to-person and computer-to-computer transferability of software throughout its life cycle.

It is intended that the software standards contained in this document be used in the future for (1) software written for the MOS development and implementation systems; (2) for all programs intended for operational implementation on the Eclipse computers either at NMC or WSFO's; and (3) for all other programs that the author intends to make part of any larger system involving other persons or that he/she intends to be maintained by others.

Any system of software standards is somewhat arbitrary. Different organizations have different standards, and textbooks do not agree. The standards set forth in this document are mainly the result of over 20 years personal experience on seven different computer systems. No one will agree with them completely; complete agreement could not be achieved for any set. These standards do have the advantage that most of the MOS software does not depart greatly from them and, therefore, several programmers within TDL are familiar with and use a large part of them.

Attitude is the main thing. I hope everyone can see the advantage, not only to TDL but for himself/herself also, of using these standards.

2. GENERAL PROGRAMMING STANDARDS

The programming language to be used for the IBM 360/195 is FORTRAN IV. S/360 assembler is to be used only in exceptional cases for subroutines where optimization is crucial; in this case, within program documentation must include an equivalent FORTRAN code. Other languages should not be used. Assembly language may be needed at times on the Data General Eclipse to perform calls to system routines that can't be called by FORTRAN.

Documentation Block - Every program and subroutine must start with a documentation block following the outline in the Appendix. Starting column convention is imposed to promote readability. Generally, in the absence of specific guidelines, standard typing rules should be used in punching comments.

Program Name - The first line should be the subroutine name starting in Col. 7. If it is a main program, substitute a Comment Card with the program name as shown in the Appendix. For main programs in the MOS system, use the naming convention in TDL Office Note 75-2.

Date, Programmer, Organization, Computer - Maintaining the exact date is not important; it is not to be used, for example, as the date the routine was added to the library. The month and year is sufficient. Starting in Col. 10, the date, the programmer's name, TDL, and the computer system the program was written for are put on the second line separated by three spaces. An extra line or two can be used here, if appropriate, to indicate modification dates, etc., but should be kept to a minimum.

Purpose - The next line should contain the word PURPOSE starting in Col. 10. Following that will be a short paragraph explaining the purpose of the routine. This need not be extensive, as details can be placed in the program writeup in TDL Office Note 75-2. Start all lines in this paragraph in Col. 14.

Data Set Use - After the paragraph on purpose, the next line should contain DATA SET USE starting in Col. 10. Listed below this line will be data set names followed by a brief explanation of them (see Appendix). The explanation should state whether they are input, output, or internal. If no data sets are used by this routine, put NONE on the card following DATA SET USE. It is not necessary to include the conventional FORTRAN use of FT05, FT06, and FT07.

Variables - The statement following those explaining data set use should contain the word VARIABLES starting in Col. 10. Following that, most, if not all, variables used in executable statements in the program should be defined in the format shown in the Appendix. The equal sign should be in Col. 23 followed and preceded by one space. (Standardization here will allow duplication of cards and use in other routines using the same variables.

Note that those variables appearing only in COMMON need not be defined.) All lines except the one defining the variable start in Col. 25 unless some further indention seems appropriate.

List all variables in the subroutine call sequence, if any, first and in order. No other ordering is mandatory, but some logical sequence would be good. For instance, list variables in COMMON in order, or list variables in order of first use in the routine, or alphabetize. For those variables in the call sequence, place at the end of the comment either (INPUT), (OUTPUT), (INPUT-OUTPUT) or (WORK AREA) to indicate its use in the subroutine. Also if the type of the variable is other than INTERGER*4 or REAL*4, indicate its type, e.g., (LOGICAL*1).

<u>General Comments</u> - Usually other comments will be more appropriate if placed within the code.

<u>In-Line Documentation</u> - In-line documentation should be provided at appropriate points in the program. It is recommended that at least 10% of the lines be comments. The comments are used to explain the code and should be subordinate to it; therefore, <u>start all comments in Col. 10</u>. One should expect to "read" the code with explanation by the comments, rather than vice versa.

<u>Punching</u> - Decks should be punched on an IBM 029 punch or one producing the equivalent bit pattern.

<u>Program Termination</u> - Normal termination should produce a termination code of zero. A statement "XXX COMPLETED" where XXX is the program name is also useful. Any other stop than normal termination should:

- Produce an error code corresponding to the statement number of the stop (i.e., use "STOP 1013" if the closest statement number to that stop is 1013).

- If the stop is in a subroutine, an error statement which includes the subroutine name should be printed (i.e., "STOP IN XXX AT YYY", where XXX is the subroutine name and YYY is the stop number would be satisfactory for a rare and difficult to explain stop--a better explanation should be used for stops that might occur more frequently).

- It is many times better to return an error code from a subroutine rather than terminate; this should be done if the user may want to exercise his judgment of how to treat the error. However, if the error is unrecoverable or if in the judgment of the author of the subroutine it would definitely be a mistake to continue, the stop should be in the subroutine. This protects the user, who might forget to check the error code.

3. FORTRAN PROGRAMMING STANDARDS

These standards are not aimed primarily at optimization, although optimization is mentioned in connection with some items. Several documents are available on optimization and can be referred to; unfortunately, these documents do not usually indicate how important a particular item is--one may be very important, another insignificant--and may apply differently to different compiler options and to different computer systems.

Length of Programs - should seldom be over 150 lines of code, exclusive of comments; less than 100 lines is much better. A specific maximum size is not as important as convenient program structure, but if a programmer finds his/her program units are consistently over 100 lines, he/she should seek advice on how to best reduce the length.

Documentation Block - should be placed first in case of a main program (and include the program name) or immediately following the subroutine name.

Declarative and Data Statements - should immediately follow the documentation block. The order is not completely specified, but COMMON, DIMENSION, TYPE, EQUIVALENCE, and DATA statements should come first and in that order.

Type Statements - should not be used unless the type is "unusual"--that is, do not use TYPE statements to dimension REAL*4 or INTEGER*4 variables.

Common Blocks - all COMMON blocks should be labelled (unless some unusual circumstance can justify otherwise). The programmer should try to use unique names rather than, say, "BLOCK1". A good name for a block in a main program would be the program name itself.

For program efficiency, it is better to have one large COMMON block than several smaller ones. Also, within the block, the variables with the larger dimensions should be placed last. Again, these are not rules never to be broken, but program efficiency should be considered.

Equivalence Statements - Equivalencing of variables weakens optimization whenever such variables appear in executable statements. Equivalencing should be avoided if its only purpose is to save a few tens of locations.

Variable Naming - The FORTRAN predefined specification of integer and real variables must be followed--INTEGER(I-N), REAL(A-H,O-Z).

Top Down Coding - should generally be followed. That is, insofar as possible without lengthening the program or using subroutines for things that are more easily or efficiently done within the routine itself, the logic should start at the top and flow to the bottom. Repeats of a set of instructions, whether by a DO loop or by branching, is, of course, necessary.

Statement Labels - should always start in Col. 2., no matter how many digits they contain.

Number only those statements to which reference is made (i.e., only those it is necessary to number).

Some logical numbering sequence must be followed. Some possibilities are:

(a) The numbers range from 1 through 9999 and be in sequence.

(b) The numbers always contain 4 digits and be in sequence.

(c) The primary numbering system start at 100 or above and end at 999, but, upon revision, when it is necessary to insert more numbers than space has been provided for, a fourth digit is added. Since all numbers start in Col. 2, they "appear" to be in order even though 1115 comes between 111 and 112.

Although (c) may seem at first glance to be more complicated, it is really very simple and workable; most of the MOS system programs follow this numbering sequence.

Statement Format - All statements (except comments) will start in Col. 7. Continuation statements, identified in sequence by 1-9 and then A-J in Col. 6, should be indented by at least 5 spaces unless there is a reason to do otherwise (e.g., this may be inconvenient in a FORMAT statement).

Statements should not include blanks unless it improves readability. Establish a pattern and stick with it. Examples as used in MOS programs are:

```
      SUBROUTINE INTR(P,BY,BX,BB)

      DIMENSION SAVE(2,2),P(61,81)

      EQUIVALENCE (P(1,1),NPK(1)),(X,Y)

      COMMON/M400/VRBL1(10),VRBL2(10),VRBL3(100),
     1          VRBL4(1000)

      CALL RDMOSH(N,NPK,NWDS,NROWS,NCOLS,JDATE,NERR)
```

```
PRINT 121,KDATE(MT),JDATE

GO TO 110

IF(JDATE-KDATE(MT))110,160,150

STOP 151

BACKSPACE NT

KCOL2=15

IF(KUT.EQ.1)KCOL2=6

DO 200 J=1,NSTA

RLONG=YP(J)

GO TO(15,16,17),KUT

FORMAT(' 'A4,4F6.1,3XI6,2I4,' ERROR 1')

X=IB(4)+IA(6)+3*(K+IC(J)**4)+M/N

SAVE(2,1)=SAVE(1,1)
```

Continuation Lines - should be denoted by the sequence of numbers 1 through 9, then A through J. (Occasionally, it may be desirable to start the sequence with 2 rather than 1 in DATA statements.)

Continue Statements - CONTINUE statements should be used only where necessary, which is normally at the end of DO loops. Do not follow the procedure recommended by some texts of always branching to a continue.

End each DO loop with a CONTINUE statement even though this is not logically necessary. This serves the purpose of notifying the "reader" that this is the end of a DO loop, and probably aids computer optimization. For nested DO loops, end each loop with a separate CONTINUE (unless the nest is extremely short and will be executed infrequently) to aid computer optimization as well as "readability."

Format Statements - will be numbered in sequence along with other statements. A FORMAT statement should immediately follow the first I/O statement which refers to it.

I/O Device Reference - in subroutines should usually be by integer variable name, preferably passed through the argument list. Main programs and very specialized subroutines that would never be used by more than one calling program may use integer constants for I/O device reference.

Variable Dimensions - The IBM 360/195 allows the use of 1 as the dimension for variables provided to the subroutine through its argument list, even though the variable may be indexed to any desired value within the subroutine. However, the dimension, if it is to be treated as variable in the subroutine, should also be carried as an argument and used (e.g., DIMENSION VARBL(NT)). Transfer to other systems may require use of the variable dimension. (With multiple dimensions, all except the last must be known.)

Subroutine Call Sequence - Constants should not appear in the argument list--define a variable and use it instead of a constant. Some order for arguments can be specified, although exceptions will occur and not all possibilities are covered:

- If a data set reference number is provided, put it first.
- Other input to the routine should follow.
- Variables used for both input and output or work area should then follow.
- Output variables, ending with an error code (if any) and finally the * return to a statement number (if any) should come last.
- Variable dimensions for an array or arrays should follow the last array name in which they are used. Multiple dimensions for an array should occur in the same sequence as they occur in the Dimension Statement.

Subroutine Entry Points - If a call to a subroutine entry point other than the subroutine name is used, put a comment at that point (or at least at the first use of the entry point) indicating the name of the subroutine being entered.

End of File and Error Checks - should be made when reading data sets.

Error Codes - The error code variable should be given a value of zero for the "no error" condition. Codes describing a data set reading error or End of File condition should be 1 and 2 respectively.

Indexing Variables with Multiple Dimensions - If convenient, use the longer dimension(s) of a variable first, and in nested DO loops index the first dimension first. This is, in general, considerably more efficient.

Output - Output to be printed should be arranged in an easy-to-read format. For instance, line up columns of numbers. Also identify values printed in well-understood terms or in terms of variables defined in the program writeup.

<u>Diagnostic Print Statements</u> - It is a nice feature, especially for programs which are to be run in an operational environment, to put diagnostic (checkout) print statements under control of an integer variable which is defined in a DATA statement. That is, give the variable a value of zero to omit print and a 1 to activate the print. In this way, a simple change to one DATA card can eliminate considerable work in putting in and taking out such statements.

4. ECLIPSE CONSIDERATIONS

The following items highlight differences between the Eclipse and the 360/195. They are not all "standards," but should be helpful in making the transition to the Eclipse.

- Variables used in data statements must also be in labeled common blocks.
- When a routine is abnormally terminating, an explicit print should be done. RDOS does not type the number in a STOP XXX statement.
- System and utility calls return a one for a successful completion. However, we should use zero for a successful completion, the same as on the 360/195.
- Any variables which are computed and used only in a subroutine should be placed in common if their value is needed when reentering the subroutine. If not in common, they are not available on reentry. The common block for these variables need not appear in other routines.
- Octal constants may be expressed by appending K to the constant (e.g. 377K).
- Subroutine and common block names must be unique within the first <u>five</u> characters.
- Formatted I/O is very costly in both CPU time and core space. Jobs that will be operational on the Eclipse system should use sequential, random, or binary I/O.
- The Eclipse compiler does not optimize, so more care may be needed in writing efficient code, especially since most of the Eclipse programs will be "operational."

5. PROGRAM DOCUMENTATION

Not all programs or individual subroutines need be documented outside the code itself. However, the within-program documentation is not intended to be a substitute for program writeups. All development programs that may have use by other than the author should be documented according to the style in TDL O.N. 75-2 and put on the W4LIB library. This documentation could pertain to a main program and all its subroutines, to an individual subroutine, or to a subroutine and its specialized subroutines. A subroutine that might be used by someone outside the specific context for which it was written should be documented and loaded on W4LIB as a separate member. However, if its use outside the context for which it was written is remote, it need not be documented except as part of a larger program unit.

The format of documentation in TDL O.N. 75-2 is mandatory for IBM/195 programs.

ACKNOWLEDGMENTS

Although considerably different in detail, these standards were developed along the same lines as the NOAA/NESS publication "Operations Data Processing Facility Computer Software Standards and Guidelines" (1976). The assistance of Mr. J. E. Poulliott, who headed the task team that developed the NESS standards, and Mr. M. P. Waters, III is gratefully acknowledged. I also thank Mr. G. H. Hollenbaugh, Mr. T. D. Bethem, and Ms. M. M. Heffernan for detailed comments and suggestions.

REFERENCES

Glahn, H. R., 1974: The TDL MOS development system IBM 360/195 version. TDL Office Note 74-14, National Weather Service, NOAA, Department of Commerce, 127 pp.

______, G. W. Hollenbaugh, and F. T. Globokar, (Editors), 1975: Computer programs for the MOS development system IBM 360/195 version. TDL Office Note 75-2, National Weather Service, NOAA, Department of Commerce, 446 pp.

NESS, 1976: Operations data processing facility computer software standards and guidelines. National Environmental Satellite Service, NOAA, U.S. Department of Commerce, 40 pp.

```
C     PROGRAM M400
C        MARCH 1979    GLAHN    TDL    IBM 360/195
C        PURPOSE
C            TO ANALYZE SEVERAL FIELDS ON SUM COMPUTATIONAL GRID.
C            PROGRAM BASED ON GLAHN ANALYSIS FOR SAM (1971), GRAYSON AND
C            BERMOWITZ ANALYSIS FOR SUM (1973), BERMOWITZ ANALYSIS FOR
C            ADVECTION PACKAGE CLAM (1973) AND CHARBA ANALYSIS PACKAGE
C            (1979).
C        DATA SET USE
C            FT01 - TDL HOURLY ARCHIVE (INPUT)
C            FT08 - TDL LFM LARGE GRID ARCHIVE (INPUT).  NEEDED ONLY IF
C                   FIRST GUESS IS TO BE FROM LFM.
C            FT10 - SCRATCH FILE (INPUT - OUTPUT)
C            FT20 - COMPLETED ANALYSES (OUTPUT)
C        VARIABLES
C                NVAR = NUMBER OF DIFFERENT VARIABLES TO BE ANALYZED.
C            KTYPE(L) = TYPE OF ANALYSIS TO PERFORM (L = 1 TO MAX OF 15)
C                       NOTE THAT KTYPE(L) IS THE INDEX K FOR ASSOCIATED
C                       VARIABLES ER1( ,K), ER2(K), R( ,K), B( ,K),
C                       NPRT( ,K), AND JPRT( ,K).  IN THIS WAY, EACH
C                       VARIABLE TO BE ANALYZED (SUCH AS SLP) HAS IT'S
C                       OWN POSITION AND VALUES CAN BE ASSIGNED IN
C                       COMMON.
C                        1 = U-WIND (M/S)
C                        2 = V-WIND (M/S) WIND IS ANALYZED AS A VECTOR
C                            WHEN KTYPE( ) = 1.  KTYPE( ) = 2 IS NEVER
C                            READ IN.
C                        3 = DEW POINT (F)
C                        4 = TEMPERATURE (F)
C                        5 = SEA LEVEL PRESSURE (MB)
C                        6 = 6-H PRECIP (INCHES)
C                        7 = CEILING (100'S OF FT)
C                        8 = OPAQUE SKY COVER (TENTHS)
C                        9 = VISIBILITY (MI)
C                       10 = (NOT USED)
C                       11 = MDR (CODE)
C                       12 = SATURATION DEFICIT
C                       13 = (NOT USED)
C                       14 = (NOT USED)
C                       15 = (NOT USED)
C            KCOL1(K) = COLUMN IN HOURLY DATA MATRIX TO FIND VARIABLE
C                       KTYPE(K), K = 1, NVAR
C         IFLDID(J,K) = 3-WORD FIELD ID, (J = 1, 3), (K = 1, NVAR),
C                       FOR OUTPUT.
C            KDATE( ) = UP TO 50 DATE-TIMES TO DO ANALYSIS FOR.
C                       YR, MO, DA, HR.  HR = 0, 23 AS IT IS ON HOURLY
C                       TAPES.
C               NDATE = NUMBER OF DATE-TIMES READ INTO KDATE( ).
C                  MT = LOOP INDEX FOR NDATE DATE-TIMES.
C            KSKIP( ) = BEFORE WRITING ANALYSES, PROGRAM WILL SKIP
C                       DOWN PAST DATE-TIME KSKIP(4) AND ANALYSIS WITH
C                       ID'S KSKIP(1), KSKIP(2), AND KSKIP(3).  IF
C                       KSKIP(4) = 0, NO SKIPPING IS DONE.
C             DSTA(J) = STATION CALL LETTERS J = 1, NSTA.
C               XP(J) = STATION LATITUDE, LATER HORIZONTAL GRID
C                       POSITION.
C               YP(J) = STATION LONGAIUDE, LATER VERTICAL GRID POSITION.
C             DATA(J) = DATA TO ANALYZE.
C            DATA1(J) = NOT USED EXCEPT WHEN KTYPE( ) = 1 OR 12.  WHEN
C                       KTYPE( ) = 1, DATA( ) CONTAINS U-WIND AND
C                       DATA1( ) V-WIND.  WHEN KTYPE( ) = 12, DATA( )
C                       CONTAINS SCALED SATURATION DEFECIT AND DATA1( )
C                       CONTAINS WEATHER VARIABLE.
```

```
        LTAG(J) = DENOTES USE OF DATA CORRESPONDING TO DSTA(J).
                  +2 = NOT USED FOR ANY PURPOSE.
C                 +1 = PERMANENTLY DISCARDED FOR THE VARIABLE
C                      BEING ANALYZED.
C                  0 = USE ON CURRENT PASS THROUGH DATA.
C                 -1 = DO NOT USE ON THIS PASS.
C     YNP, XNP = POLE POSITION ON SUM COMPUTATIONAL GRID.
C                POSITION (1,1) AT LOWER LEFT, POLE AT (87,37)
C        MY,MX = SIZE OF GRID (MY,MX) = (61,81)
C           KV = LOOP INDEX FOR VARIABLE BEING ANALYZED.
C       P( , ) = FIELD HOLDING FIRST GUESS AND ANALYSIS.
C     ER1(J,K) = ERROR CRITERIA FOR PASS (J = 1, 8) AND VARIABLE
C                (K = 1, 15).  IF OBSERVATION IS DIFFERENT FROM
C                CURRENT ANALYSIS BY MORE THAN ER1( , ), IT IS
C                NOT USED ON THIS PASS.  ALSO IF ER1(J,K) = 0,
C                IT MEANS CHECK IS NOT PERFORMED ON THIS PASS.
C       ER2(K) = ERROR CRITERIA FOR VARIABLE (K = 1, 15) WHEN
C                DOING BUDDY CHECK ON FIRST PASS.  IF THE
C                DISTANCE BETWEEN THE STATION AND ITS BUDDY IS
C                LE 1 GRID LENGTH, THE DIFFERENCE IN OBS IS
C                CHECKED AGAINST ER2( ).  WHEN THE DISTANCE IS
C                GT 1 GRID LENGTH, THE DIFFERENCE IN OBS DIVIDED
C                BY THE DISTANCE IS CHECKED AGAINST ER2( ).
C                ER2( ) IS NOT USED FOR SLP (K = 5) -- SEE
C                ER3( ).
C       ER3(L) = ERROR CRITERIA FOR BUDDY CHECK FOR SLP FOR
C                PRESSURE INCREMENT L (L = 1, 13).
C       R(J,K) = RADIUS OF INFLUENCE FOR PASS J (J = 1, 8) AND
C                VARIABLE K (K = 1, 15).
C    NPRT(J,K) = 1 (2) (3) FOR PRINTING OF ANALYSIS ON SMALL
C                (LARGE) (BOTH SMALL AND LARGE) GRID AFTER PASS
C                J (J = 1, 8) AND VARIABLE K (K = 1, 15).
C                ZERO FOR NO PRINTING.
C    JPRT(J,K) = SAME AS ABOVE EXCEPT FOR SMOOTHED ANALYSIS.
C     NPASS(K) = NUMBER OF PASSES TO PERFORM FOR VARIABLE
C                (K = 1, 15).   (NOT REALLY NEEDED AS NTYPE( , )
C                CAN = 0 WHICH MEANS SKIP THIS PASS.)
C    IGUESS(K) = SPECIFIES TYPE OF FIRST GUESS FOR VARIABLE K
C                (K = 1, 15)
C                 0 = NONE
C                 1 = CONSTANT VALUE FROM GUESS(K)
C                 2 = LFM FIELD
C   NTYPE(J,K) = TYPE OF CORRECTION FOR PASS J (J = 1, 8) AND
C                VARIABLE K (K = 1, 15).
C                   0 MEANS SKIP THIS PASS
C                   1 MEANS W = 1
C                   2 MEANS W = (R**2 - D**2)/(R**2 + D**2)
C                   3 MEANS SAME AS 2 EXCEPT SUM OF WEIGHTS IN
C                     DENOMINATOR.
C                   4 MEANS SET THE GRIDPOINT VALUE TO THE CLOSEST
C                     STATION VALUE.
C       B(J,K) = SMOOTHING PARAMETER FOR PASS J (J = 1, 8) AND
C                VARIABLE K (K = 1, 15).  B( , ) = 0 MEANS NO
C                SMOOTHING.  THIS CAN BE USED FOR ALL CORRECTION
C                TYPES NTYPE( , ).
C   TITLE(J,K) = TITLE FOR VARIABLE K, J = 1, 33 WORDS.
C     CNST1(J) = GRID POINT CONSTANT NO. 1 FOR VARIABLE
C                (K = 1, 15).
C     CNST2(J) = GRID POINT CONSTANT NO. 2 FOR VARIABLE
C                (K = 1, 15).  NEW VALUE = OLD VALUE * CNST2 ( )
C                + CNST1( )
C     CNST3(J) = GRID POINT CONSTANT NO. 3, PLACE TO START
C                CONTOURS.
C     CNST4(J) = GRID POINT CONSTANT NO. 4, CONTOUR INTERVAL.
```

```
      COMMON/M400/MT,NSTA,XNP,YNP,MX,MY,KV,KSKIP(4),ER3(13),KTYPE(15),
     1            KCOL1(15),ER2(15),NPASS(15),CNST1(15),CNST2(15),
     2            CNST3(15),CNST4(15),IGUESS(15),IFLDID(3,15),
     3            LFM1(3,15),LFM2(3,15),KDATE(50),ER1(8,15),R(8,15),
     4            NPRT(8,15),JPRT(8,15),NTYPE(8,15),P(8,15),
     5            NFGES(24,3,2),TITLE(33,15),LTAG(1500),DSTA(1500),
     6            XP(1500),YP(1500),DATA(1500),DATA1(1500),NPK(15429)
      READ(5,110)NVAR
  110 FORMAT(I4)
      DO 125 K=1,NVAR
      READ(5,110)KTYPE(K)
      KVT=KTYPE(K)
      READ(5,120)(ER1(J,KVT),J=1,8),(NTYPE(J,KVT),J=1,8),
     1            (B(J,KVT),J=1,8),(R(J,KVT),J=1,8),(NPRT(J,KVT),J=1,8),
     2            (JPRT(J,KVT),J=1,8),IGUESS(KVT)
  120 FORMAT(8F4.0/8I4/8F4.0/8F4.0/8I4/8I4/I4)
  125 CONTINUE
      CALL RDDATA(5,KDATE,50,DATA,7,'(7I10)',NDATE,99999999)
C         READS UP TO 50 DATE-TIMES TO DO ANALYSIS FOR.
      READ(5,130)KSKIP
  130 FORMAT(2XZ8,2XZ8,2XZ8,I10)
      PRINT 140,NDATE,(KDATE(K),K=1,NDATE)
  140 FORMAT('1ANALYSES FOR'I3,' DATES'//(10I10))
      PRINT 141,(KSKIP(K),K=1,4)
  141 FORMAT('0OUTPUT SKIPPING PARAMETERS'//2XZ8,2XZ8,2XZ8,I10)
      DO 145 K=1,NVAR
      KVT=KTYPE(K)
      PRINT 142,KTYPE(K),(ER1(J,KVT),J=1,8),(NTYPE(J,KVT),J=1,8),
     1            (B(J,KVT),J=1,8),(R(J,KVT),J=1,8),(NPRT(J,KVT),J=1,8),
     2            (JPRT(J,KVT),J=1,8),IGUESS(KVT)
    2 FORMAT('0VARIABLE'I6/
     1        ' ER1      '8F6.1/
     2        ' NTYPE    '8I6/
     3        ' B        '8F6.1/
     4        ' R        '8F6.1/
     5        ' NPRT     '8I6/
     6        ' JPRT     '8I6/
     7        ' IGUESS   'I6)
  145 CONTINUE
C         READS OUTPUT TAPE SKIPPING PARAMETERS.
      CALL SKIP
C         SUBROUTINE SKIP SKIPS OUTPUT TAPE TO DESIRED PLACE.
      DO 160 MT=1,NDATE
      DO 150 KV=1,NVAR
      CALL SFCDTA
C         DATA FOR ANALYSIS ARE RETRIEVED FROM TDL HOURLY OBS TAPES.
C         LTAG( ) SET.
      CALL FSTGES
C         FIRST GUESS FIELD NOW IN P( , ) IF NEEDED.
      CALL BCD
  150 CONTINUE
  160 CONTINUE
      PRINT 170
  170 FORMAT(///'0M400 COMPLETED')
      STOP
      END
      SUBROUTINE FINDLT(LIST,KT,ITEM,M,*)
C        APRIL 1979   HOLLENBAUGH   TDL   IBM 360/195
C        PURPOSE
C            TO SEARCH FOR ITEM IN AN ORDERED LIST AND RETURN TO THE
C            CALLING PROGRAM THE LOCATION OF ITEM IN LIST.
C        DATA SET USE
C            NONE
C        RETURNS
C              RETURN - ITEM FOUND IN ORDERED LIST AND LOCATION
C                       RETURNED.
C            RETURN 1 - ITEM NOT FOUND IN ORDEDED LIST.  TAKE
C                       APPROPRIATE ACTION.
C        VARIABLES
C             LIST( ) = ORDERED ARRAY TO BE SEARCHED.  (INPUT)
C                  KT = SIZE OF ORDERED LIST.  (INPUT)
C                ITEM = VARIABLE TO BE FOUND IN ORDERED LIST.  (INPUT)
C                   M = LOCATION OF ITEM IN ORDERED LIST.  (OUTPUT)
      DIMENSION LIST(KT)
      K=1
      L=KT
 110  M=SHFTR(L-K,1)+K
C        ABOVE STATEMENT SAME AS M=(L-K)/2+K, BUT FASTER.
      IF(L-M.LT.4)GO TO 125
      IF(LIST(M)-ITEM)115,140,120
 115  K=M
      GO TO 110
 120  L=M
      GO TO 110
 125  DO 130 M=K,KT
      IF(LIST(M)-ITEM)130,140,135
 130  CONTINUE
 135  RETURN 1
 140  RETURN
      END
      SUBROUTINE RDMOL(ITAPE,MH,NIWDS)
C        MARCH 1979    GLAHN    TDL    IBM 360/195
C        ADAPTED FROM 1979 BETHEM, GLAHN, AND GLOBOKAR READI
C        PURPOSE
C            READS PACKED DATA FROM GRID-PT TAPES.
C        DATA SET USE
C            ITAPE - TDL GRID-POINT TAPE ARCHIVE (INPUT)
C        VARIABLES
C               ITAPE = INPUT DATA SET NUMBER. (INPUT)
C               MH( ) = ARRAY TO READ GRID-POINT FIELD INTO. (OUTPUT)
C               NIWDS = SIZE OF ARRAY TO PUT INTO MH( ). (INPUT)
      DIMENSION MH(NIWDS)
      DATA NERR/0/
 200  READ(ITAPE,ERR=230,END=235)MH
      GO TO 240
 230  NERR=NERR+1
      IF(NERR-500)231,2300,233
 2300 PRINT 2301
 2301 FORMAT(///'0ADDITIONAL READING ERRORS WILL NOT BE PRINTED'//)
      GO TO 233
 231  WRITE(6,232)(MH(K),K=1,12)
 232  FORMAT(///'0READING ERROR ',12(1XZ8)//' PROGRAM WILL SKIP THIS REC
     1ORD AND CONTINUE'//)
 233  READ(ITAPE,ERR=230,END=235)
C        ABOVE STATEMENT NECESSARY TO TRANSFER BAD DATA FROM BUFFER.
      GO TO 200
 235  WRITE(6,237)ITAPE
 237  FORMAT('0END OF FILE ON GRID POINT TAPE FT'I2)
      GO TO 200
 240  RETURN
      END
 BLOCK DATA
    MARCH 1979   GLAHN    TDL    IBM 360/195
    PURPOSE
        TO FURNISH DATA TO M400
 COMMON/M400/MT,NSTA,XNP,YNP,MX,MY,KV,KSKIP(4),ER3(13),KTYPE(15),
1              KCOL1(15),ER2(15),NPASS(15),CNST1(15),CNST2(15),
2              CNST3(15),CNST4(15),IGUESS(15),IFLDID(3,15),
3              LFM1(3,15),LFM2(3,15),KDATE(50),ER1(8,15),R(8,15),
4              NPRT(8,15),JPRT(8,15),NTYPE(8,15),B(8,15),
5              NFGFS(24,3,2),TITLE(33,15),LTAG(1500),DSTA(1500),
6              XP(1500),YP(1500),DATA(1500),DATA1(1500),NPK(15429)
 DATA YNP,XNP/87.,37./
 DATA MY,MX/61,81/
 DATA IGUESS/15*2/
 DATA KCOL1 /  10,   11,    7,   12,   21,   13,    3,  1,    5,  20,    0,
2               7,    0,    0,    0/
 DATA ER2   /  0.,   0.,  20.,  20.,  17.,  10*9./
 DATA ER3   / 17.,  16.,  15.,  14.,  13.,  11.,   9., 7.,   5.,  3.,   3.,
2              2.,   2./
 DATA CNST1 /  0.,   0.,   0.,   0.,   0.,   0.,   0., 0.,   0.,  0.,   0.,
2              0.,   0.,   0.,   0./
 DATA CNST2 / 10.,  10.,  10.,  10.,  10.,  10.,100.,10.,100.,  10.,  10.,
2             10.,  10.,  10.,  10./
 DATA CNST3 / 25.,  25.,  50.,  50.,  30.,  30.,  10.,50.,  50.,  10.,  10.,
2            200.,  10.,  10.,  10./
 DATA CNST4 /  0.,   0.,   0.,   0.,   0.,   0.,   0., 0.,   0.,  0.,   0.,
2              0.,   0.,   0.,   0./
 DATA NPASS /   8,    8,    8,    8,    8,    8,    8,  8,    8,   8,    8,
2               8,    8,    8,    8/
 DATA NFGES / -1,  -1,  -1,  -1,   0,   0,   0,   0,   0,   0,   0,   0,
2              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
3             12,  12,  12,  12,   0,   0,   0,   0,   0,   0,   0,   0,
4              0,   0,   0,   0,  12,  12,  12,  12,  12,  12,  12,  12,
5             12,  12,  12,  12,   6,   6,   6,   6,   6,   6,  12,  12,
6             12,  12,  12,  12,   6,   6,   6,   6,   6,   6,  12,  12,
7             -1,  -1,  -1,  -1,  -1,  -1,  -1,   0,   0,   0,   0,   0,
8              0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
9             12,  12,  12,  12,  12,  12,  12,   0,   0,   0,   0,   0,
A              0,   0,   0,   0,   0,   0,   0,  12,  12,  12,  12,  12,
B             12,  12,  12,  12,  12,  12,  12,  12,  12,  12,  12,  12,
C             12,  12,  12,  12,  12,  12,  12,  12,  12,  12,  12,  12/
 DATA ER1   /    60.,     30.,     20.,     15.,      0.,      0.,     0.,     0.,
2                60.,     30.,     20.,     15.,      0.,      0.,     0.,     0.,
3                40.,     20.,     10.,      5.,      0.,      0.,     0.,     0.,
4                40.,     20.,     10.,      5.,      0.,      0.,     0.,     0.,
5              14.0,    11.6,     6.0,     3.0,      0.,      0.,     0.,     0.,
6                 0.,      5.,      3.,      3.,      0.,      0.,     0.,     0.,
7                 0.,    100.,     50.,     50.,      0.,      0.,     0.,     0.,
8                 0.,      7.,      5.,      4.,      0.,      0.,     0.,     0.,
9                 0.,      5.,      3.,      2.,      0.,      0.,     0.,     0.,
A                     110.,   110.,    50.,    30.,     0.,     0.,     0.,     0.,
B                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
C                      60.,    65.,    60.,    55.,     0.,     0.,     0.,     0.,
D                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
E                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
F                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0./
 DATA NTYPE /            3,      3,      3,      2,      0,      0,      0,      0,
2                        3,      3,      3,      2,      0,      0,      0,      0,
3                        3,      3,      3,      2,      0,      0,      0,      0,
4                        3,      3,      3,      2,      0,      0,      0,      0,
5                        2,      3,      3,      3,      0,      0,      0,      0,
6                        3,      3,      3,      2,      0,      0,      0,      0,
7                        3,      3,      3,      2,      0,      0,      0,      0,
8                        3,      3,      3,      2,      0,      0,      0,      0,
9                        3,      3,      3,      2,      0,      0,      0,      0,
A                        3,      3,      3,      2,      0,      0,      0,      0,
B                        1,      1,      1,      1,      0,      0,      0,      0,
C                        2,      2,      3,      3,      0,      0,      0,      0,
D                        1,      1,      1,      1,      0,      0,      0,      0,
E                        1,      1,      1,      1,      0,      0,      0,      0,
F                        1,      1,      1,      1,      0,      0,      0,      0/
 DATA B      /          0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
2                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
3                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
4                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
5                       0.,     5.,     1.,     2.,     0.,     0.,     0.,     0.,
6                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
7                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
8                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
9                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
A                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
B                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
C                       0.,     0.,     1.,     4.,     0.,     0.,     0.,     0.,
D                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
E                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
F                       0.,     0.,     0.,     0.,     0.,     0.,     0.,     0./
 DATA R      /         8.0,    3.0,    2.0,    1.0,     0.,     0.,     0.,     0.,
2                      8.0,    3.0,    2.0,    1.0,     0.,     0.,     0.,     0.,
3                      8.0,    5.0,    2.5,    1.0,     0.,     0.,     0.,     0.,
4                      8.0,    5.0,    2.5,    1.0,     0.,     0.,     0.,     0.,
5                      8.0,    5.0,    2.5,    1.0,     0.,     0.,     0.,     0.,
6                      8.0,    3.0,    2.0,    1.0,     0.,     0.,     0.,     0.,
7                      8.0,    3.0,    1.0,    0.0,     0.,     0.,     0.,     0.,
8                      8.0,    3.0,    1.0,    0.0,     0.,     0.,     0.,     0.,
9                      8.0,    3.0,    1.0,    0.0,     0.,     0.,     0.,     0.,
A                      8.0,    4.0,    2.0,    1.0,     0.,     0.,     0.,     0.,
B                      8.0,    0.0,    0.0,    0.0,     0.,     0.,     0.,     0.,
C                      8.0,    0.0,    0.0,    0.0,     0.,     0.,     0.,     0.,
D                      0.0,    0.0,    0.0,    0.0,     0.,     0.,     0.,     0.,
E                      0.0,    0.0,    0.0,    0.0,     0.,     0.,     0.,     0.,
F                      0.0,    0.0,    0.0,    0.0,     0.,     0.,     0.,     0./
 DATA TITLE /' U-W','IND ',31*'    ',
2            ' V-W','IND ',31*'    ',
3            ' DEW',' POI','NT  ',30*'    ',
4            ' TEM','PERA','TURE',30*'    ',
5            ' SEA',' LEV','EL P','RESS','URE ',28*'    ',
6            ' SUR','FACE',' PRE','SSUR','E   ',28*'    ',
7            ' 6-H','R PR','ECIP',' AMO','UNT ',28*'    ',
     8        ' CEI','LING',31*'     ',
     9        ' VIS','IBIL','ITY ',30*'     ',
     A        ' OPA','QUE ','SKY ','COVE','R   ',28*'     ',
     B        ' MLR',32*'     ',
     C        ' SAT','URAT','ION ','DEFI','CIT ',28*'     ',
     D       33*'     ',
     E       33*'     ',
     F       33*'     '/
      DATA LFM1   / Z03009000, Z00000000, Z20009000,
     2              Z03109000, Z00000000, Z20009000,
     3              Z01000800, Z00271081, Z00000000,
     4              Z01100800, Z00271081, Z00000000,
     5              Z00808000, Z00000000, Z00000000,
     6              Z00808100, Z00000000, Z00000000,
     7              Z00000000, Z00000000, Z00000000,
     8              Z00000000, Z00000000, Z00000000,
     9              Z00000000, Z00000000, Z00000000,
     A              Z00000000, Z00000000, Z00000000,
     B              Z00000000, Z00000000, Z00000000,
     C              Z05809100, Z00823585, Z20009000,
     D              Z00000000, Z00000000, Z00000000,
     E              Z00000000, Z00000000, Z00000000,
     F              Z00000000, Z00000000, Z00000000/
      DATA LFM2   / Z00000000, Z00000000, Z00000000,
     2              Z00000000, Z00000000, Z00000000,
     3              Z00808000, Z00000000, Z00000000,
     4              Z00808000, Z00000000, Z00000000,
     5              Z00000000, Z00000000, Z00000000,
     6              Z00000000, Z00000000, Z00000000,
     7              Z00000000, Z00000000, Z00000000,
     8              Z00000000, Z00000000, Z00000000,
     9              Z00000000, Z00000000, Z00000000,
     A              Z00000000, Z00000000, Z00000000,
     B              Z00000000, Z00000000, Z00000000,
     C              Z00000000, Z00000000, Z00000000,
     D              Z00000000, Z00000000, Z00000000,
     E              Z00000000, Z00000000, Z00000000,
     F              Z00000000, Z00000000, Z00000000/
      END
```